Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

направление подготовки: 09.03.01 - «Информатика и вычислительная техника»

**О Т Ч Е Т**

**по лабораторной работе “Классы” №6**

**по дисциплине**

**«Основы алгоритмизации и программирования» семестр 2**

Выполнил студент гр. ИВТ-21-1б

Ипатов Дмитрий Сергеевич

Проверил:

Ст. Преподаватель кафедры ИТАС

Яруллин Д.В.

(оценка) (подпись)

г. Пермь-2022

**Постановка задачи:**

1. Определить класс-контейнер.
2. Реализовать конструкторы, деструктор, операции ввода-вывода, операцию присваивания.
3. Перегрузить операции, указанные в варианте.
4. Реализовать класс-итератор. Реализовать с его помощью операции последовательного доступа.
5. Написать тестирующую программу, иллюстрирующую выполнение операций.

**Анализ задачи:**

АТД - тип данных, определяемый только через операции, которые могут выполняться над соответствующими объектами безотносительно к способу представления этих объектов.

АТД включает в себя абстракцию как через параметризацию, так и через спецификацию.

**Абстракция через параметризацию** может быть осуществлена так же, как и для процедур (функций); использованием параметров там, где это имеет смысл.

**Абстракция через спецификацию** достигается за счет того, что операции представляются как часть типа.

Для реализации АТД необходимо, во-первых, выбрать представление памяти для объектов и, во-вторых, реализовать операции в терминах выбранного представления.

Примером абстрактного типа данных является класс в языке С++.

Если элементы контейнера не упорядочены, то добавление и удаление элементов обычно выполняется в начале и в конце контейнера.

Итератор можно реализовать как класс, представляющий такой же набор операций. В С++ итератор реализуется как класс, который имеет такой же интерфейс, как и указатель для совместимости с массивами.

**Код программы:**

#include <iostream>

using namespace std;

class Iterator

{

friend class List;

protected:

int\* elem;

public:

Iterator() { elem = 0; }

Iterator(const Iterator& it) { elem = it.elem; }

bool operator== (const Iterator& it) { return elem = it.elem; }

bool operator!=(const Iterator& it) { return elem != it.elem; }

void operator++() { ++elem; }

void operator--() { --elem; }

int& operator\* () const { return\*elem; }

};

class List

{

protected:

int size;

int\* data;

Iterator beg;

Iterator end;

public:

List(int s, int k = 0);

List(const List& p);

~List();

int& operator[](int index);

List operator+(const int k);

List operator+(const List&p);

List& operator=(const List& p);

int& operator()();

friend ostream& operator<<(ostream& out, const List& p);

friend istream& operator>>(istream& in, List& p);

Iterator first() { return beg; }

Iterator last() { return end; }

};

List::List(int s, int k)

{

size = s;

data = new int[size];

for (int i = 0; i < size; i++)

data[i] = k;

beg.elem = &data[0];

end.elem = &data[size];

}

List::List(const List& p)

{

size = p.size;

data = new int[size];

for (int i = 0; i < size; i++)

data[i] = p.data[i];

beg = p.beg;

end = p.end;

}

List::~List()

{

delete[]data;

data = 0;

}

List List::operator+(const List& p)

{

int l;

if (size < p.size) l = p.size;

else l = size;

List temp1(l);

for (int i = 0; i < size; i++)

temp1.data[i] += data[i];

for (int i = 0; i < p.size; i++)

temp1.data[i] += p.data[i];

return temp1;

}

List& List::operator=(const List& p)

{

if (this == &p) return \*this;

size = p.size;

if (data != 0) delete[]data;

data = new int[size];

for (int i = 0; i < size; i++)

data[i] = p.data[i];

beg = p.beg;

end = p.end;

return \*this;

}

int& List::operator[](int index)

{

if (index < size) return data[index];

else cout << "ОШИБКА ДОСТУПА! (index>size)";

}

List List::operator+(const int k)

{

List temp(size);

for (int i = 0; i < size; i++)

temp.data[i] = data[i] + k;

return temp;

}

int& List::operator()()

{

return size;

}

ostream& operator<<(ostream& out, const List& p)

{

for (int i = 0; i < p.size; i++)

out << p.data[i] << " ";

return out;

}

istream& operator>>(istream& in, List& p)

{

for (int i = 0; i < p.size; i++)

in >> p.data[i];

return in;

}

int main()

{

setlocale(LC\_ALL, "rus");

List a(5);

cout << a << '\n';

cin >> a;

cout << a;

a[0] = 20;

cout << '\n';

cout << a << '\n';

cout << \*(a.first()) << endl;

Iterator i = a.first();

++i;

cout << \*i << endl;

for (i = a.first(); i != a.last(); ++i) cout << \*i << " ";

List b(10);

b = a + 1;

cout << '\n' << b << '\n';

List c(10);

c = c + 1;

cout << b+c;

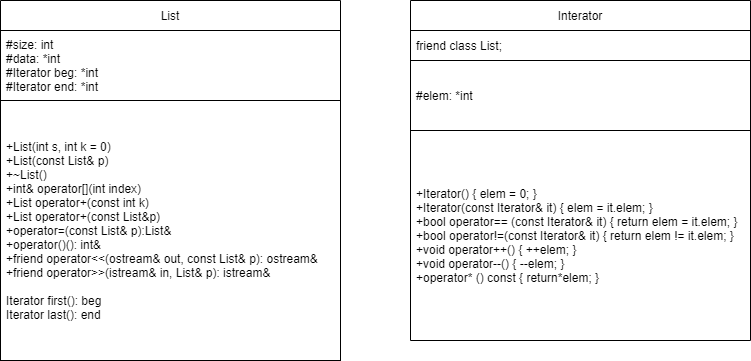
/\*List p(4);

p[10] = 15;\*/

return 0;

}

**UML:**

****

**Контрольные вопросы:**

* 1. Что такое абстрактный тип данных? Привести примеры АТД.

АТД - тип данных, определяемый только через операции, которые могут выполняться над соответствующими объектами безотносительно к способу представления этих объектов.

Пример: контейнер

* 1. Привести примеры абстракции через параметризацию.

Примером абстракции через параметризацию является концепция «функция», передаваемые параметры которой являются формальными, а фактические связываются с ними в момент использования такой абстракции.

* 1. Привести примеры абстракции через спецификацию.

Абстракции через спецификацию реализуются, например, в виде библиотечных функций, широко используемых программистами.

* 1. Что такое контейнер? Привести примеры.

Контейнер — это набор некоторого количества обязательно однотипных элементов, упакованных в контейнер определённым образом. Простейшим прототипом контейнера в классическом языке C++ является массив.

* 1. Какие группы операций выделяют в контейнерах?

Вставка, удаление элементов, получение данных, хранящихся в ячейке, перемещение по элементам, поиск элементов, объединение контейнеров, специальные операции, которые зависят от типа контейнера.

* 1. Какие виды доступа к элементам контейнера существуют? Привести примеры.

Доступ к элементам может осуществляться: последовательно, напрямую и ассоциативно. Доступ напрямую подразумевает доступ к элементу по его индексу ( Пример: a[10]); доступ ассоциативно выполняется также по индексу, но индексом будет не номер элемента, а его содержимое (Пример:a[“word”]); при последовательном доступе перемещение осуществляется от элемента к элементу контейнера и содержит набор операций последовательного доступа (Пример: v.first, v.last, v.next, v.prev и т.д)

* 1. Что такое итератор?

По факту это класс, который позволяет перемещаться по элементам контейнера из функции main.

* 1. Каким образом может быть реализован итератор?

Итератор может быть реализован как часть класса-контейнера в виде набора методов v.first, v.last, v.next, v.prev, v.skip и т.д, так и как класс, представляющий такой же набор операций.

* 1. Каким образом можно организовать объединение контейнеров?
  + Простое сцепление двух контейнеров: в новый контейнер попадают сначала элементы первого контейнера, потом второго, операция не коммутативна.
  + Объединение упорядоченных контейнеров, новый контейнер тоже будет упорядочен, операция коммутативна.
  + Объединение контейнеров как объединение множеств, в новый контейнер попадают только те элементы, которые есть хотя бы в одном контейнере, операция коммутативна.
  + Объединение контейнеров как пересечение множеств, в новый контейнер попадают только те элементы, которые есть в обоих контейнерах, операция коммутативна.
  + Для контейнеров-множеств может быть еще реализована операция вычитания, в контейнер попадают только те элементы первого контейнера, которых нет во втором, операция не коммутативна.
  + Извлечение части элементов из контейнера и создание нового контейнера. Эта операция может быть выполнена с помощью конструктора, а часть контейнера задается двумя итераторами.
  1. Какой доступ к элементам предоставляет контейнер, состоящий из элементов «ключ-значение»?

Ассоциативный доступ.

* 1. Как называется контейнер, в котором вставка и удаление элементов выполняется на одном конце контейнера?

Стек

* 1. Какой из объектов (a,b,c,d) является контейнером?
     1. int mas=10;
     2. int mas;
     3. struct {char name[30]; int age;} mas;
     4. int mas[100];
  2. Какой из объектов (a,b,c,d) не является контейнером?

a. int a[]={1,2,3,4,5};

1. int mas[30];
2. struct {char name[30]; int age;} mas[30];
3. int mas;
4. Контейнер реализован как динамический массив, в нем определена операция доступ по индексу. Каким будет доступ к элементам контейнера?

Прямой доступ

1. Контейнер реализован как линейный список. Каким будет доступ к элементам контейнера?

Последовательный доступ